

Cyclone Codes

Christian Schindelhauer¹, Andreas Jakoby², and Sven Köhler³

¹University of Freiburg, Germany, schindel@informatik.uni-freiburg.de

²Bauhaus-Universität Weimar, Germany, andreas.jakoby@uni-weimar.de

³University of Freiburg, Germany, koehlers@informatik.uni-freiburg.de

May 4, 2016

Abstract

We introduce Cyclone codes which are rateless erasure resilient codes. They combine Pair codes with Luby Transform (LT) codes by computing a code symbol from a random set of data symbols using bitwise XOR and cyclic shift operations. The number of data symbols is chosen according to the Robust Soliton distribution. XOR and cyclic shift operations establish a unitary commutative ring if data symbols have a length of $p - 1$ bits, for some prime number p . We consider the graph given by code symbols combining two data symbols. If $n/2$ such random pairs are given for n data symbols, then a giant component appears, which can be resolved in linear time. We can extend Cyclone codes to data symbols of arbitrary even length, provided the Goldbach conjecture holds.

Applying results for this giant component, it follows that Cyclone codes have the same encoding and decoding time complexity as LT codes, while the overhead is upper-bounded by those of LT codes. Simulations indicate that Cyclone codes significantly decreases the overhead of extra coding symbols.

1 Introduction

Currently, video streaming is the dominant source of traffic in the Internet. Here, most packets are delivered best effort, which means that packets can be erased anytime. The missing packet needs to be resent, which costs a round trip time. In result, the video may be halted and rebuffered if not enough packets are buffered before-hand. This can be avoided if the network layer (and link layer) provides real-time behavior. For this, IPv6 provides special quality-of-service flags in order to prioritize media packets. The best approach so far is forward error correction with so-called erasure codes.

This is only one of many applications for erasure codes, where additional redundant packets are added such that the original packets can be reconstructed from the remaining packets. RAID disks and long distance satellite communications are other prominent examples.

Our Results By combining several techniques from previous erasure resilient codes, namely Pair codes [20], Circulant Cauchy codes [23], and Luby Transform (LT) codes [15], we present a novel erasure resilient code system called Cyclone codes.

The number of data symbols combined by XOR and cyclic shift operations is chosen according to the Robust Soliton distribution. XOR and cyclic shift operations establish a unitary commutative ring if the data symbol has length $p - 1$ for some prime number $p \geq 3$. We consider the graph induced by code symbols describing two data symbols. If $n/2$ such random pairs are given for n data symbols, then the giant component appears [9], which can be resolved using these operations. We can extend Cyclone codes to data symbols of arbitrary even length provided the Goldbach conjecture holds, which has been shown for any reasonable symbol length.

Cyclone codes are rateless, non-systematic, memoryless erasure codes which share their asymptotic coding and decoding time and asymptotic coding overhead with LT codes. Yet, simulations show that it considerably improves the coding overhead by at least 40% compared to LT codes.

2 Model

We assume a sender and receiver connected by a faulty communication channel in the *packet erasure model*: (a) packets are delivered in the correct order and (b) individual packets are either correctly delivered or the receiver is notified about their loss.

Data symbols (the input to the erasure code) are denoted by x_1, x_2, \dots, x_n , where n is the number of available data symbols. The code symbols (the output of the erasure code) are denoted by y_1, y_2, \dots, y_m , where m denotes, depending on context, the number of code symbols transmitted or received. Data and code symbols are w -bit sequences.

For all three erasure codes discussed in this paper, the code symbols are linear combinations of several data symbols. The indexes of the data symbols used as well as the coefficients are random numbers. We assume that sender and receiver have access to a shared source of randomness, i.e., the receiver can reproduce the random numbers drawn by the sender. This can be achieved, for example, by using a pseudo-random generator with the same initialization on both sides. We do not discuss the issue how both sides can agree on the same initialization. Also, our analysis neglects questions regarding the time needed to generate the (pseudo-)random numbers or the reliability of their randomness. Note that the packet erasure model allows the receiver to reproduce the random number used in the computation of every code symbol. Alternatively, the sender may embed these parameters in each packet.

3 Related Work

For a short history of coding theory we refer to the excellent survey of [7]. The notion of an erasure channel was introduced in [8]. The standard method for erasure codes are Reed-Solomon codes [21]. In [3], Cauchy matrices have been proposed for the encoding. That results in a systematic code where data and code symbols can be used to reconstruct all data symbols. Symbols are elements in a Galois field and the number of finite field operations to encode k code symbols is $\mathcal{O}(kn)$ and decoding all data given k code symbols and $n - k$ data symbols takes $\mathcal{O}(kn)$ finite field operations. In [14] an efficient implementation of this systematic code is described using the word length of a processor. In [23] the circuit complexity of the Cauchy matrix approach was improved to $(3 + o(1))knw$ operations for encoding k code symbols and $9knw$ XOR bit operations for decoding from k code symbols for symbol size w , which is also the asymptotic lower bound for perfect systematic erasure codes [2].

In [23] circulant matrices have been used with the word length w , where $w + 1$ is a prime number and 2 is a primitive root for $w + 1$. If w does not have this property, w can be partitioned into sub-codes with word size w_i , such that $w = \sum_i w_i$ and each $w_i + 1$ has this property.

Circulant matrices are equivalent to cyclotomic rings introduced by Silverman [25] and can be used for fast multiplication in finite fields. They are used in [10, 13] for a small complexity arithmetic circuits for finite fields. Other applications are VLSI implementations [26], fast multiplication [6], quantum bit operations [1], and public key cryptography [19].

The computational complexity of erasure codes can be improved, if one receives more code symbols than data symbols. An attempt to overcome the limitation of the large coding and decoding complexity are Tornado codes [5, 16], where data symbols are encoded by XORs described by a cascaded graph sequence combining bipartite sub-graphs. For Tornado codes an overhead of $1 + \epsilon$ code symbols were produced, which could be coded and decoded in time $\mathcal{O}(n \log(1/\epsilon))$. Starting from this observations, Luby presented LT codes in his seminal paper [15]. They use a random set of data symbols combined by XOR into the code symbols. The underlying Robust Soliton distribution has two parameters c and δ , which has been optimized for small n in [12]. Luby has shown that, with high probability, LT codes have an overhead of only $\mathcal{O}(\sqrt{n} \log n)$ code symbols to allow decoding. Furthermore the coding and

decoding time is at most $\mathcal{O}(n \log n)$ parallel XOR operations (multiplied by w for the number of bits in the underlying symbols).

Based on LT codes, raptor codes have been introduced [24]. For any given $\epsilon > 0$ it is possible to recover from $n(1 + \epsilon)$ code symbols to the original n data symbol with a complexity of $\mathcal{O}(\log(1/\epsilon))$ operations for encoding a code symbol and $\mathcal{O}(k \log(1/\epsilon))$ operations to recover k data symbols. Except Cyclone codes, introduced here, LT codes are the only way to implement raptor codes. For this, LT codes have been analyzed in more detail [17] to improve the behavior of raptor codes.

4 Pair Codes

In [20] an erasure code called *Pair codes* is introduced. The name reflects that each code symbol is a linear combination of two distinct data symbols, i.e.,

$$y_\ell := f_{\ell,1}x_{i_{\ell,1}} + f_{\ell,2}x_{i_{\ell,2}} \quad .$$

The encoder chooses the indexes $i_{\ell,1}, i_{\ell,2} \in [1, n]$ and the non-zero coefficients $f_{\ell,1}, f_{\ell,2}$ uniformly at random. The coefficients as well as data and code symbols are seen as elements of $\mathbb{GF}(2^w)$ with $w \geq 2$, a finite field of size 2^w . To compute a code symbol, one addition and two multiplications in $\mathbb{GF}(2^w)$ are required, leading to a total encoding complexity of m additions and $2m$ multiplications.

The decoder builds the graph G_y with the vertex set $\{x_1, x_2, \dots, x_n\}$. For each code symbol y_ℓ , the graph G_y contains an edge $e_\ell = (x_{i_{\ell,1}}, x_{i_{\ell,2}})$. The decoder uses the following operations:

- *The Single Rule:* Given an edge e_ℓ connecting a decoded $x_{i_{\ell,1}}$ with an undecoded $x_{i_{\ell,2}}$, we can decode $x_{i_{\ell,2}}$ by $x_{i_{\ell,2}} = f_{\ell,2}^{-1}(y_\ell + f_{\ell,1}x_{i_{\ell,1}})$.
- *Parallel Edge Resolution:* Given two parallel edges e_k and e_ℓ , i.e., $i_{k,1} = i_{\ell,1}$ and $i_{k,2} = i_{\ell,2}$, decoding $x_{i_{\ell,2}}$ is possible by solving the system

$$y_k = f_{k,1}x_{i_{\ell,1}} + f_{k,2}x_{i_{\ell,2}} \quad \text{and} \quad y_\ell = f_{\ell,1}x_{i_{\ell,1}} + f_{\ell,2}x_{i_{\ell,2}}$$

if the coefficients of y_k and y_ℓ are linearly independent. This yields

$$x_{i_{\ell,2}} = (f_{k,1}f_{\ell,2} + f_{k,2}f_{\ell,1})^{-1}(f_{k,1}y_\ell + f_{\ell,1}y_k) \quad .$$

- *Edge Contraction:* Given two adjacent edges e_k and e_ℓ , i.e., $i_{k,2} = i_{\ell,1}$, the corresponding code symbols

$$y_k = f_{k,1}x_{i_{k,1}} + f_{k,2}x_{i_{\ell,1}} \quad \text{and} \quad y_\ell = f_{\ell,1}x_{i_{\ell,1}} + f_{\ell,2}x_{i_{\ell,2}}$$

can be transformed into a new edge $e' = (x_{i_{k,1}}, x_{i_{\ell,2}})$ and a code symbol y' with

$$y' = f_{\ell,1}y_k + f_{k,2}y_\ell = f_{k,1}f_{\ell,1}x_{i_{k,1}} + f_{k,2}f_{\ell,2}x_{i_{\ell,2}} \quad .$$

- *The Double Rule:* Given a cycle (e_1, e_2, \dots, e_t) in G_y , applying the edge contraction rule $t - 2$ times to the path (e_2, e_3, \dots, e_t) yields an edge e' parallel to e_1 . Then, the parallel edge resolution allows to decode the corresponding data symbols.

One can easily verify that for the single rule one needs to perform one inversion, two multiplications, and one addition per data symbol. For the parallel edge resolution one needs to execute one inversion, five multiplications, and two additions. The edge contraction only needs four multiplications and one addition. Considering circuit complexity additions and multiplications can be implemented by linear size and constant depth circuits if unbounded MOD2 gates are available. If we cannot use a look up table for the inversion, then current implementations, like in [25], use linear depth circuits. Thus to improve the decoding complexity one has either to minimize the number of inversions or one has to look for a coding system, that uses specific values for $f_{i,j}$ which allow efficient inversion.

To improve the complexity of the double rule, one has to modify the graph G_y whenever one perform the double rule in such a way the the resulting graph reduces its diameter. Within a perfect scenario G_y is a star graph.

The decoder applies the single rule whenever possible. Using this rule, any connected component of G_y can be decoded, as soon as its first data symbol is known. As unencoded code symbols are never sent, the double rule is applied to initiate decoding. It can be applied to any connected component of G_y which contains a cycle. We note that if the double rule is applied to some cycle C , the parallel edge resolution may fail since the coefficients are linearly dependent. This is the case if only if $f_{k,1}f_{\ell,2} = f_{k,2}f_{\ell,1}$ which happens with probability $\frac{1}{2^w-1}$ as $f_{k,1}$ is uniformly distributed. In this case we can discard one edge of C and the corresponding code symbol, as it is a linear combination of the code symbols corresponding to the other edges of C .

To discuss the number of cycles of a connected component we use the notion of *excess* of a component, which is the difference between its numbers of edges and its number of nodes. The above yields the following result:

Corollary 1. *The expected number of applications of the double rule per connected component until it succeeds is $1 + \frac{1}{2^w-2}$. The probability of decoding a connected component U of G_y is $1 - \left(\frac{1}{2^w-1}\right)^{\text{excess}(U)+1}$.*

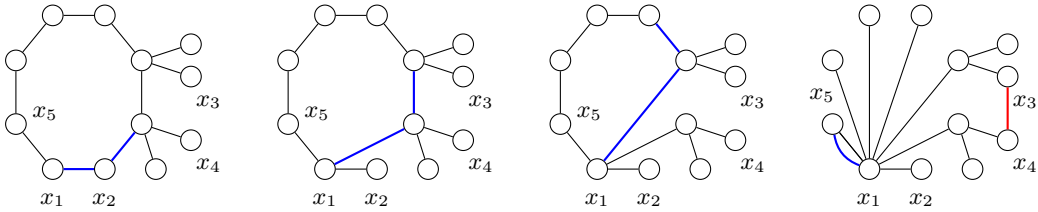


Figure 1: Influence of graph modifications on the cycle length.

As illustrated in Figure 1, performing the double rule and removing the correct redundant edge replaced a cycle with a star graph. While Pair codes have nearly linear coding and decoding complexity, they suffer from the coupon collector problem, as we see in Section 9.

5 Luby Transform Codes

With Luby Transform codes [15], each code symbol is the bitwise XOR of k_ℓ distinct data symbols, where $k_\ell \in [1, n]$ is chosen from a *special* random distribution and the k_ℓ distinct indexes $i_{\ell,j}$ are chosen uniformly at random. We call y_ℓ a clause of size k_ℓ and write

$$y_\ell = \sum_{j=1}^{k_\ell} x_{i_{\ell,j}} .$$

A basic principle of LT codes is that the size of clauses can be reduced as more and more data symbols are decoded. Namely, if $x_{i_{\ell,d}}$ for some $d \in [1, k_\ell]$ has been decoded, then y_ℓ can be replaced with the following clause of size $k_\ell - 1$:

$$y' = y_\ell + x_{i_{\ell,d}} = \sum_{j \in \{1, \dots, k_\ell\} \setminus \{d\}} x_{i_{\ell,j}} .$$

The decoder greedily reduces the size of clauses until the size of a clause is one. Then a data symbol has been successfully decoded. However, decoding cannot start unless the encoder sends clauses of size one. Unlike Pair codes, the decoder does not exploit linearly independent clauses of size 2 or larger.

The encoding and decoding complexity as well as the overhead depends heavily on the distribution of the clause sizes. Luby discusses two distributions.

The *Ideal Soliton distribution* is given by the probability mass function $\rho(k)$:

$$\rho(k) := \begin{cases} \frac{1}{n} & \text{for } k = 1 \\ \frac{1}{k(k-1)} & \text{for } k \in \{2, 3, \dots, n\} \end{cases} .$$

In the average this distribution produces one clause of size one per n code symbols and every second code symbol is a clause of size two. The expected clause size is H_n , where $H_n \approx \ln(n)$ is the n -th harmonic number.

Because of the small number of unary clauses and the small probability that all data symbol are addressed, the Ideal Soliton distribution works poorly [15]. Thus, Luby introduces the *Robust Soliton distribution* μ . It is a combination of ρ and another distribution τ . Let $R = c \cdot \ln(n/\delta)\sqrt{n}$, where $c > 0$ and $\delta \in (0, 1)$ are tunable parameters, and define

$$\tau(k) := \begin{cases} \frac{R}{kn} & \text{for } k = 1, 2, \dots, \lfloor \frac{n}{R} - 1 \rfloor \\ \frac{R \ln(R/\delta)}{n} & \text{for } k = \lfloor \frac{n}{R} \rfloor \\ 0 & \text{otherwise} \end{cases} .$$

The Robust Soliton distribution is defined as

$$\mu(k) := \frac{\rho(k) + \tau(k)}{\beta} \quad \text{with} \quad \beta = \sum_{k=1}^n \rho(k) + \tau(k) ,$$

where the factor of $1/\beta$ normalizes the probability mass function.

The addition of τ boosts the probability of clauses with sizes less than n/R . In particular, the expected number of clauses of size 1 is increased from 1 to about R per n clauses. The expected clause size of the Robust Soliton distribution is $\mathcal{O}(\ln(n/\delta))$ [15] and thus is asymptotically equal to the expected clause size of the Ideal Soliton distribution if δ is constant. This observation directly implies the following result for constant symbol size w :

Corollary 2. *The expected encoding and decoding complexity of an LT code is $\mathcal{O}(m \ln(n))$ XOR word operations using the Ideal Soliton distribution and $\mathcal{O}(m \ln(n/\delta))$ XOR word operations using the Robust Soliton distribution.*

Under the Robust Soliton distribution $m = n + \mathcal{O}(\sqrt{n} \log n)$ code symbols suffice to decode all data symbols with high probability, i.e., $1 - n^{-\mathcal{O}(1)}$.

6 Circulant Matrices and Cyclotomic Rings

We avoid time or memory consuming multiplication operations in finite fields by following the approach of cyclotomic rings [25]. These are equivalent to circulant matrices, on which Circulant Cauchy codes [23], a systematic perfect erasure resilient code, are based. Here, we modify this concept for general prime numbers p , while previously, for word size w , it was required that $p = w + 1$ is a prime number with 2 as a primitive root.

For such p , the cyclotomic polynomial of degree w , $\Phi(z) = z^w + \dots + z^2 + z + 1$, is irreducible. The finite field $\mathbb{GF}(2^w)$ is a sub-ring of the ring of polynomials modulo $z^p - 1$, since $z^p - 1 = \Phi(z)(z - 1)$. In order to get efficient multiplication and division operations for $\mathbb{GF}(2^w)$ each input $b = (b_0, \dots, b_{w-1})$ is extended by a so-called *ghost bit* $b_w = 0$. Then, all operations are done in the ring with the extended ghost bit basis and retransformed for being output. This way, Silverman [25] reduces the complexity of school multiplication by a factor of 2.

In [23] an equivalent approach is followed, yet only multiplications and divisions with monoms $b_i z^i$ and binomials $b_i z^i + b_j z^j$ are used. Such operations have linear bit complexity $\mathcal{O}(w)$ and all operation

except the division by a binomial can be computed in constant number of steps by a processor with word length $\Theta(w)$.

Now let $p = w + 1$ be a prime number, where $\Phi(z)$ is not necessarily irreducible. We give now a formal description of our operations and the underlying ring $R_p = \{0, 1\}^p$. We define for $x = (x_0, \dots, x_{p-1})$, $y = (y_0, \dots, y_{p-1})$ the addition and the multiplication. For $k \in \{0, \dots, p - 1\}$ we have

$$(x + y)_k = (x_k + y_k) \bmod 2, \quad (x \cdot y)_k = \sum_{i=0}^{p-1} x_i y_{k-i \bmod p} \bmod 2.$$

From now on addition on bits is always modulo 2, i.e., the XOR operation, and $\bar{b} := 1 + b$ for $b \in \{0, 1\}$. We name the constants $\mathbb{O} := (0, \dots, 0)$, $\mathbb{I} := (1, 0, \dots, 0)$, and $\mathbb{D} := (0, 1, 0, \dots, 0)$, and $\mathbb{A} := (1, \dots, 1)$, each of length $w + 1$. Therefore, $\mathbb{D}^i = (0^i, 1, 0^{w-i})$. So, the multiplication with monomials \mathbb{D}^i in R_p , as well its inverse operation, is a cyclic shift operation. The multiplication with binomials $\mathbb{D}^i + \mathbb{D}^j$ needs $w + 1$ XOR operations. The multiplication with polynomials with k non-zero terms takes $(w + 1)(k - 1)$ XOR operations. We now deal with the problem of dividing by binomials.

We use the following transformation between external w -bit representation and internal p -bit *Ghost Bit Basis* representation R_p for $x_i \in \{0, 1\}$ and $p = w + 1$:

$$\text{pad}(x_0, x_1, \dots, x_{w-1}) := (x_0, \dots, x_{w-1}, 0) \quad (1)$$

$$\text{unpad}(x_0, x_1, \dots, x_w) := (x_0 + x_w, x_1 + x_w, \dots, x_{w-1} + x_w). \quad (2)$$

Under these transformations we get the following elements in the ghost bit base for each $x \in \{0, 1\}^w$:

$$G(x) := \{(x_0, \dots, x_{w-1}, 0), (\bar{x}_0, \dots, \bar{x}_{w-1}, 1)\} \quad (3)$$

such that $\text{pad}(u) \subseteq G(u)$ and $\text{unpad}(G(x)) = x$. For two sets S_1, S_2 we define

$$S_1 + S_2 := \{s_1 + s_2 \mid s_1 \in S_1, s_2 \in S_2\} \quad S_1 \cdot S_2 := \{s_1 \cdot s_2 \mid s_1 \in S_1, s_2 \in S_2\}$$

The equivalency class G is closed in R_p under addition and multiplication ($\exists!$ denotes unique existential quantification).

Lemma 3. *We have for all $x, y \in \{0, 1\}^w$*

$$G(x) + G(y) = G(x + y) \quad (4)$$

$$\exists! z \in \{0, 1\}^w : G(x) \cdot G(y) = G(z) \quad (5)$$

Proof. For easier notion let $x0$ denote the w -bit *symbol* x followed by 0 and $x1$ denote x followed by 1. Equation (4): Recall that \bar{x} denotes the bitwise negation of the vector x and the bitwise XOR is denoted by the addition.

$$\begin{aligned} G(x) + G(y) &= \{x0, \bar{x}1\} + \{y0, \bar{y}1\} \\ &= \{(x + y)0, \overline{(x + y)}1\} \\ &= G(x + y) \end{aligned}$$

Equation (5): First we consider the monomial $y = 2^i = (0^i 1 0^{w-i})$ and get

$$\begin{aligned} G(x) \cdot G(2^i) &= \{(x_0, \dots, x_{w-1}, 0), \overline{(x_0, \dots, x_{w-1}, 0)}\} \cdot \{\mathbb{D}^i, \overline{\mathbb{D}^i}\} \\ &= \{(x_0, \dots, x_{w-1}, 0), \mathbb{A} + (x_0, \dots, x_{w-1}, 0)\} \cdot \{\mathbb{D}^i, \mathbb{A} + \mathbb{D}^i\} \\ &= \{(x_0, \dots, x_{w-1}, 0) \cdot \mathbb{D}^i, \mathbb{A} \cdot \mathbb{D}^i + (x_0, \dots, x_{w-1}, 0) \cdot \mathbb{D}^i, \\ &\quad (x_0, \dots, x_{w-1}, 0) \cdot \mathbb{A} + (x_0, \dots, x_{w-1}, 0) \cdot \mathbb{D}^i, \\ &\quad \mathbb{A} \cdot \mathbb{A} + (x_0, \dots, x_{w-1}, 0) \cdot \mathbb{A} + \mathbb{A} \cdot \mathbb{D}^i + (x_0, \dots, x_{w-1}, 0) \cdot \mathbb{D}^i\}. \end{aligned}$$

Note that for even w we have

$$\begin{aligned} \mathbb{A} \cdot \mathbb{A} &= \mathbb{A} \\ \mathbb{A} \cdot \mathbb{D}^i &= \mathbb{A} \\ (x_0, \dots, x_{w-1}, 0) \cdot \mathbb{A} &= \begin{cases} \mathbb{O} & \text{if the number of } j \text{ with } x_j = 1 \text{ is even} \\ \mathbb{A} & \text{otherwise.} \end{cases} \end{aligned}$$

Thus we get

$$\begin{aligned} G(x) \cdot G(2^i) &= \{(x_0, \dots, x_{w-1}, 0) \cdot \mathbb{D}^i, \mathbb{A} + (x_0, \dots, x_{w-1}, 0) \cdot \mathbb{D}^i\} \\ &= \{(x_{w-i+1}, \dots, x_{w-1}, 0, x_0, \dots, x_{w-i}), \overline{(x_{w-i+1}, \dots, x_{w-1}, 1, x_0, \dots, x_{w-i})}\} \end{aligned}$$

Hence, $|\text{unpad}(G(x) \cdot G(2^i))| = 1$. Now consider $y = (y_0, \dots, y_{w-1})$ and by using (4) we have

$$G(x) \cdot G(y) = \sum_{i=0}^{w-1} y_i \cdot G(x) \cdot G(2^i) .$$

From $|\text{unpad}(x + y)| = 1$ the claim follows by an induction over the number of non-zero components of y . \square

Multiplication with monomial and binomials can be efficiently inverted. Note that this observation cannot be extended to other polynomials unless 2 is a primitive root for $p = w + 1$.

Lemma 4. *We have for all $y \in \{0, 1\}^w$, $i, j \in \{0, \dots, w\}$, $i \neq j$*

$$\exists! x \in \{0, 1\}^w : \quad G(x) \cdot \mathbb{D}^i = G(y) \quad (6)$$

$$\exists! x \in \{0, 1\}^w : \quad G(x) \cdot (\mathbb{D}^i + \mathbb{D}^j) = G(y) \quad (7)$$

In both cases x can be computed with w bitwise XOR operations.

Proof. Equation (6): Note that the multiplication with a monomial \mathbb{D}^i , i.e., a cyclic shift by i bits, maps an element of R_{w+1} and its complement to another element and its complement. A cyclic shift of $w + 1 - i$ bits reverses this operation. Since $|\text{unpad}(G(y) \cdot \mathbb{D}^{w+1-i})| = 1$ and $|\text{unpad}(G(x) \cdot \mathbb{D}^i)| = 1$ there is no other solution than

$$x = \text{unpad}(\text{pad}(y) \cdot \mathbb{D}^{w+1-i}) .$$

Equation (7): For $x', y' \in R_{w+1}$ such that $x = \text{unpad}(x')$, $y = \text{unpad}(y')$ the equation $x'(\mathbb{D}^i + \mathbb{D}^j) = y'$ is equivalent to the equations

$$x'_{(k+i) \bmod (w+1)} + x'_{(k+j) \bmod (w+1)} = y'_k, \quad \text{for } k = 0, \dots, w$$

or equivalently

$$x'_{(k+j-i) \bmod (w+1)} = x'_k + y'_{(k-i) \bmod (w+1)}, \quad \text{for } k = 0, \dots, w .$$

We choose an element from the class $G(\text{unpad}(x'))$ by setting $x'_w = 0$ and compute

$$x'_{(w+j-i) \bmod (w+1)} = y'_{(w-i) \bmod (w+1)} .$$

Then,

$$x'_{(w+2(j-i)) \bmod (w+1)} = x'_{(w+(j-i)) \bmod (w+1)} + y'_{(w+(j-i)-i) \bmod (w+1)}$$

and by induction we can compute for all $k = 1, 2, \dots, w$

$$x'_{(w+k(j-i)) \bmod (w+1)} = x'_{(w+(k-1)(j-i)) \bmod (w+1)} + y'_{(w+(k-1)(j-i)-i) \bmod (w+1)} .$$

If $p = w + 1$ is prime then all entries of x' are determined, since we jump through all the indices of x_i with step size $(j - i)$ modulo $(w + 1)$. If we would have set $x_w = 1$ we would have received the complement of x' as solution, which is the only other solution. Further note, that in this process all equations are satisfied. Which proves that $x = \text{unpad}(x')$ is a solution and the only one. Clearly, the number of XOR-operations is $p - 1 = w$. \square

With respect to the equivalency class G , the multiplication with binomials is invertible in R_{w+1} and thus we use for the above operation the notation $x = y \cdot (\mathbb{D}^i + \mathbb{D}^j)^{-1}$ for $x, y \in R_{w+1}$.

Consequently, Cyclone codes use the ring multiplication with monomials and binomials, since these operations can be implemented as cyclic shifts and bitwise XOR operations.

7 Cyclone Codes

Given a vector of input symbols $x_1, \dots, x_n \in \{0, 1\}^w$ we produce $m \geq n$ output symbols $y_1, \dots, y_m \in \{0, 1\}^w$ using a random process. We assume that $p = w + 1$ is prime. Recall the Robust Soliton distribution μ as defined in Section 5. Each code symbol y_1, \dots, y_m is constructed as follows.

Algorithm 1 Encoding of Cyclone Codes

```

for  $\ell \leftarrow 1$  to  $m$  do
     $k_\ell \leftarrow$  randomly chosen according the probability mass function  $\mu$ 
     $i_{\ell,1}, \dots, i_{\ell,k_\ell} \leftarrow$  randomly, uniformly chosen distinct values from  $\{1, \dots, n\}$ 
     $f_{\ell,1}, \dots, f_{\ell,k_\ell} \leftarrow$  randomly, independently, uniformly chosen from  $\{0, \dots, w\}$ 
     $y_\ell \leftarrow \text{unpad} \left( \sum_{j=1}^{k_\ell} \mathbb{D}^{f_{\ell,j}} \cdot \text{pad}(x_{i_{\ell,j}}) \right)$ 
end

```

Lemma 5. *A Cyclone code symbol y_ℓ can be constructed with expected $w \cdot \mathbb{E}[k_\ell]$ XOR bit operations.*

Proof. This follows from the observation that unpadding takes w XOR operations and the sum over a clause of k elements takes $(k - 1)w$ XOR operations. The cyclic shift and the padding operations do not need any operation at all. \square

Decoding

When reading a code symbol $y_\ell \in \{0, 1\}^w$ we compute the padded version in the ghost bit representation $g_\ell = \text{pad}(y_\ell)$. Then, all computations are done in the ghost bit basis and the extra bit is only removed, when the data symbol is to be issued. So, for the ℓ -th code symbol the following information is stored $g_\ell \in R_p$, $k_\ell, i_{\ell,1}, \dots, i_{\ell,k_\ell} \in \{1, \dots, n\}$, $f_{\ell,1}, \dots, f_{\ell,k_\ell} \in \{0, \dots, w\}$. We call this a clause of size k_ℓ where the following invariant holds

$$\text{unpad}(g_\ell) = \text{unpad} \left(\sum_{j=1}^{k_\ell} \mathbb{D}^{f_{\ell,j}} \cdot \text{pad}(x_{i_{\ell,j}}) \right). \quad (8)$$

On these clauses we implement the following operations:

- *Read code symbol:* Given y_ℓ , determine $g_\ell = \text{pad}(y_\ell)$ and store all parameters of the clause.
- *Output data symbol:* Given a clause g_ℓ of size 1 (a monomial), i.e., $k_\ell = 1$, determine the output $x_{\ell,1}$:

$$x_{\ell,1} = \text{unpad}(\mathbb{D}^{p-f_{\ell,1}} \cdot g_\ell) = \text{unpad}(\mathbb{D}^{p-f_{\ell,1}} \cdot \mathbb{D}^{f_{\ell,1}} \cdot \text{pad}(x_{i_{\ell,1}})) .$$

- *Monomial reduction:* Given a monomial g_u ($k_u = 1$) and a polynomial g_v ($k_v \geq 1$) such that $i_{1,u} = i_{j,v}$, we reduce the size of g_v by one by removing $i_{j,v}$ and $f_{j,v}$ and replacing g_v with $g'_v = g_v + \mathbb{D}^{f_{j,v}-f_{1,u}} \cdot g_u$.
- *Parallel edge resolution:* Given two clauses g_u, g_v of size $k_u = k_v = 2$ with $(i_{1,u}, i_{1,v}) = (i_{2,u}, i_{2,v})$ and $f_{1,u} - f_{2,v} \not\equiv f_{1,u} - f_{2,v} \pmod{p}$, we create two monomials g_a and g_b , with $\text{unpad}(g_a) = x_{i_{1,u}}$, $\text{unpad}(g_b) = x_{i_{1,v}}$, replacing g_u, g_v as follows:

$$g_a = (\mathbb{D}^{f_{1,u}+f_{2,v}} + \mathbb{D}^{f_{1,v}+f_{2,u}})^{-1} \cdot (\mathbb{D}^{f_{2,v}} g_1 + \mathbb{D}^{f_{1,v}} g_2) \quad (9)$$

$$g_b = (\mathbb{D}^{f_{1,u}+f_{2,v}} + \mathbb{D}^{f_{1,v}+f_{2,u}})^{-1} \cdot (\mathbb{D}^{f_{2,u}} g_1 + \mathbb{D}^{f_{1,u}} g_2) \quad (10)$$

The monomials g_a and g_b replace g_u and g_v . If $f_{1,u} - f_{2,v} \equiv f_{1,v} - f_{2,u} \pmod{p}$, then one of the clauses is redundant and can be removed.

- *Edge Contraction*: Given two connected clauses g_u, g_v of size 2, i.e., $i_{1,v} = i_{2,u}$, generate a new clause g'_2 with indices $(i'_{2,u}, i'_{2,v}) = (i_{1,u}, i_{2,v})$ where $f'_{2,u} = f_{1,u} + f_{2,u}$, $f'_{2,v} = f_{1,v} + f_{2,v}$, $g'_v = \mathbb{D}^{f_{2,u}} g_u + \mathbb{D}^{f_{1,v}} g_v$. Replace g_v by g'_v .

Lemma 6. *All these clause operations need at most $4w$ XOR bit operations and preserve the invariant (8).*

Proof. This follows from the linear complexity of the padding, unpadding, addition, cyclic shift and binomial inverse operation. \square

The first three operations are equivalent to LT codes. Resolving codes this way is called the *single rule*.

Lemma 7. *Cyclone codes using only monomial reductions are equivalent to LT codes.*

Proof. Since the probability distribution for clause lengths is the same, it is sufficient to prove that the cyclic shift operations do not influence the resolution, which is a straight-forward observation. \square

Monomial reduction and parallel edge resolution can be used to resolve large sets of data symbols connected by binomials. For this we consider the subset of clauses of size 2 and the corresponding clause graph $G = (V, E)$ with $V = \{1, \dots, n\}$ and edge set of clauses of sizes 1 and 2. Loops describe clauses of size 1, where parallel edges are allowed. Recall that the excess of a connected component is defined as the difference between the number of edges and nodes in this component.

Lemma 8. *If in the clause graph G there exists a connected component with excess $c \geq 0$ and q nodes, then all data symbols of this components can be resolved with $\mathcal{O}(qw)$ XOR bit operations with probability $1 - 1/(w+1)^{c+1}$.*

Proof. If the excess is non-negative, then there exists a cycle (e_1, \dots, e_t) in this component. Using the edge contraction, we can shorten this cycle by replacing $e_1 = \{u_1, u_2\}, e_2 = \{u_2, u_3\}$ within the cycle by edge $e'_1 = \{u_1, u_3\}$. After $t-2$ edge contractions we have reduced the cycle to a cycle of length 2 consisting of two parallel edges. These two parallel edges can be resolved using the parallel edge resolution if $f_{1,1} - f_{2,2} \not\equiv f_{1,2} - f_{2,1} \pmod{p}$. Note that all operations uphold the uniform probability distribution for the cyclic shift factors. Hence, this equality does not holds with probability $\frac{1}{w+1}$. In this case we remove one edge of the considered cycle and thus we decremented excess of the component. If the new excess is still non-negative we can restart the process with a new cycle.

Otherwise, if $f_{1,1} - f_{2,2} \equiv f_{1,2} - f_{2,1} \pmod{p}$ we resolve u_1 and u_2 which allows us to resolve all data symbols of the connected components repeatedly using the monomial reduction.

Depending on the way we choose the edges within the edge contraction we can bound the number of edge contractions by the number of nodes in the components plus the number of linear dependent edges, if we transform each cycle to a star. \square

The resolution of connected components in the clause graph is called the *double rule*. The decoding algorithm applies the single and double rule as long as they are successful.

Algorithm 2 Cyclone Decoding

```
for  $\ell \leftarrow 1$  to  $m$  do
  Read code symbol  $y_\ell$ 
  repeat
    repeat
      | Apply Single Rule
    until Single Rule does not change clauses;
    repeat
      | Apply Double Rule
    until Double Rule does not change clauses;
  until code set not changed in this round;
end
```

It is not fully understood why the double rule improves efficiency. An explanation is that the clause graph is an Erdős-Renyi-Graph [9]. For $n/2$ edges a giant component of sizes $\Theta(n^{2/3})$ appears and grows dramatically for increasing number of edges, where for every random edge four elements are added to the component [4]. There it is also shown that for $n/2 + \omega(n^{2/3} \log^{1/2} n)$ edges the excess of the giant component is non-negative, which allows the decoding of this component. Understanding this process in combination of the effect on clauses larger than 2 might help to build better probability distributions for the Cyclone codes as the Robust Soliton distributions already allow. For the efficiency and coding overhead the following observation can be shown.

Theorem 9. *Cyclone codes need at most the asymptotic time complexity and coding overhead as LT codes.*

Proof. The coding overhead follows from Lemma 7, where we have seen that LT codes are included in the first three rules.

For the time complexity, we have seen that applying the pair rule adds only a linear number of operations for each decoded data symbol. Most time is still consumed by the single rules, where longer clauses are stripped from decoded data symbols. An operation which also takes place in LT codes, yet with some additional code symbols. \square

8 Fermat, Goldbach and the Word Length

All operations except the binomial division can be performed word parallel, such that w operations can be done in constant number of processor steps. It turns out that the binomial division is only needed once for each connectivity component, so its influence is marginal.

It is of particular interesting to use powers of 2 as the word length w . However, since $w + 1 = p$ is required to be a prime number, this would restrict us to Fermat primes of form $2^{2^i} + 1$. They are the only numbers of form $2^j + 1$ which can be prime. Only 5 Fermat primes are known, corresponding to word lengths 2, 4, 16, 256, and 65 536.

In [23], a small trick is introduced which generalizes the word lengths to all even positive numbers. For this we split a data symbol of length w into two separate encodings of lengths w_1 and w_2 , where $w_1 + 1$ and $w_2 + 1$ are prime numbers. This is possible if $w + 2$ can be represented as sum of two primes p_1, p_2 , which is the still open Goldbach conjecture, one of Hilbert's eight problems [11]. It is known that this is the case for $w \leq 4 \cdot 10^{14}$ [22]. Then, we send the code symbols in both encodings without overhead. However, we can only decode if both codes can be decoded, which might lead to an increased overhead, unless $w_1 = w_2$. Then we can use the same parameters, twice. Because of the probabilities depending on the word length, the best choice is to choose w_1 and w_2 of nearly equal size, which leads to the following word lengths, e.g.,

$$\begin{array}{lll} 8 & = & 4 + 4 \\ 128 & = & 58 + 70 \end{array} \quad \begin{array}{lll} 32 & = & 16 + 16 \\ 512 & = & 256 + 256 \end{array} \quad \begin{array}{lll} 64 & = & 28 + 36 \\ 1024 & = & 502 + 522 \end{array} .$$

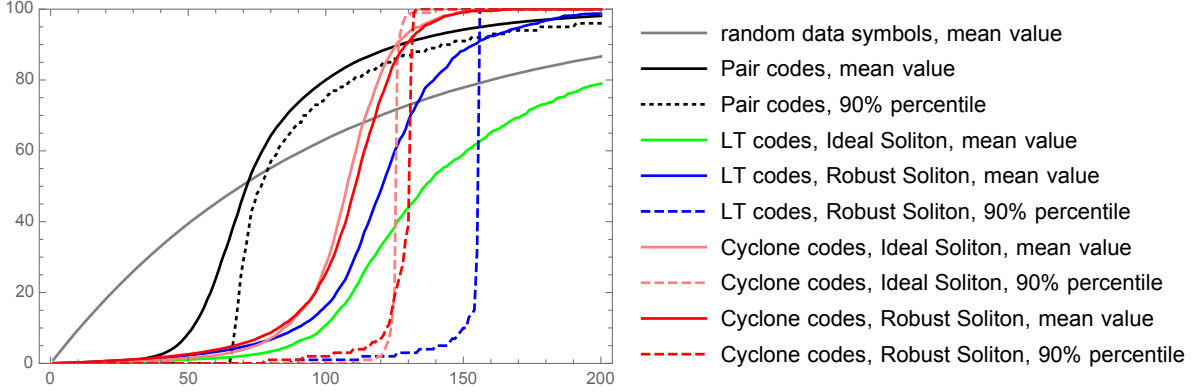


Figure 2: The average number and 90%-percentile of the number of decoded symbols with respect to the number of available fountain codes for 100 data symbols and 1000 test samples.

9 Simulations

We have run extended simulations to estimate the overhead of the Cyclone code. For this we generate a series of s code symbols and count the number of decoded data symbols. We compare Cyclone codes using the Ideal Soliton distribution and the Robust Soliton distribution for the clause size with LT codes on the same clause size distributions. Furthermore, we show the behavior of uniformly chosen random data symbols (suffering under the coupon collector problem) and Pair codes, where the pairs are chosen uniformly.

Figure 2 shows the number of decoded data symbols (vertical axis), for a growing series of s coded symbols for the above mentioned codes. The number of overall data symbols is $n = 100$. The word size is $w = 256$. For the Robust Soliton distribution we chose as parameters $\delta = 0.5$ and $c = 0.01$ [18]. The straight lines represent the average over 1000 tests. The dotted lines show the 90%-percentile of the set of decoded data symbols from 1000 runs with different random numbers. In Appendix A, we show the corresponding runs for $n = 1000$.

The probability that a data symbol is not covered after m code symbols is described by $(1 - \frac{1}{n})^m$, which can be approximated by $e^{-m/n}$ for large enough n . Hence, the expected number of available symbols follows the function $n(1 - e^{-m/n})$, as the simulations clearly show. For Pair codes we see only few decoded symbols before $m = n/2$. At $m = n/2$ the random clause graph shows the appearance of the giant component of size $\Theta(n^{2/3})$. So, the probability of a cycle in such a component tends towards 1 and the decoding starts. In the long run, Pair codes suffer from a reduced coupon-collector problem, where the upper limit of the function is $n(1 - e^{-2m/n})$, since the probability that a data node is not covered in the clause graph is $(1 - \frac{2}{n})^m$.

It is known that LT codes using the Ideal Soliton distribution do not perform well. For $n = 100$ and $n = 1000$ it behaves worse than sending random data symbols. Hence, it is quite surprising that for $n = 100$ a Cyclone code performs even better than Cyclone codes with respect to the Robust Soliton distributions. In the median 118 code symbols (overhead 18%) are sufficient and for only 10% of the samples more than 136 code symbols were necessary. Cyclone codes with Robust Soliton distribution needs more than 138 code symbols for only 10% of the samples and 125 in the median. The overhead is calculated as the relative number of extra code symbols in order to decode the data symbols, i.e., $\frac{m-n}{n} = \frac{m}{n} - 1$. For LT codes it is known that they have considerable overhead for such small number of data symbols. It is recommended to use at least $n \geq 10000$.

Figure 3 takes this into account by increasing the number of data symbols $n = 2^i$ for $i = 1, \dots, 18$, i.e., $n = 2, 4, 8, \dots, 262144$, we have performed 1000 tests with a series of fountain code symbols, where we stopped the test each time at the m -th code symbol, when all data symbols could be computed. Then, we computed the overhead $m/n - 1$. For random data symbols, Pair codes, LT codes with Ideal and Robust Soliton distribution ($\delta = 0.5$, $c = 0.01$), and Cyclone codes with Ideal and Robust Soliton

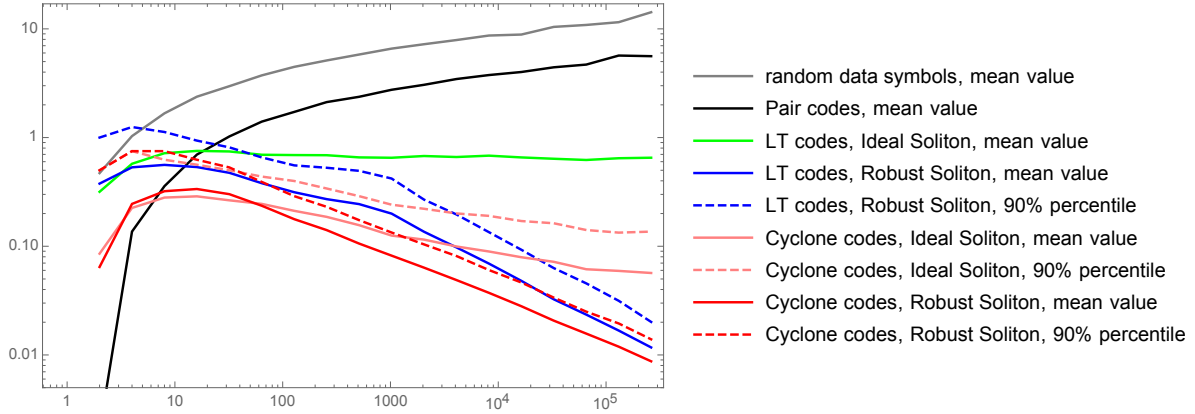


Figure 3: A log-log-plot of the average (and 90%-percentile) relative overhead of extra code symbols to be sent in different fountain codes for increasing number of data symbols.

distribution we plotted the average overhead and the 90%-percentile in a log-log-plot. The word size was chosen as $w = 256$.

For small $n < 8$ Pair codes performs best, then the Cyclone code with Ideal Soliton distribution takes over until for $n \geq 128$ the Cyclone code with Robust Soliton consistently outperforms all other displayed codes. For $n = 8192$ Cyclone codes with a Robust Soliton distribution have a median overhead of 3.4%, an average overhead of 3.9%, a standard deviation of 1.5%, and a 90%-percentile of 5.9%. The corresponding values for LT codes are a median overhead of 4.5%, an average overhead of 5.8%, a standard deviation of 3.5%, and a 90%-percentile of 10.3%.

10 Conclusions

Cyclone codes combine LT codes [15] with Pair codes [20] and decrease the message overhead without increasing the coding and decoding complexity. The coding overhead is $(m - n)/n$ where m the number of code symbols to be received to decode n data symbols. By the nature of XOR operations, LT codes can not benefit from the giant component in the random clause graph, i.e., the graph described by code symbols combining two data symbols. There, every cycle corresponds to redundant code symbols, which can be immediately discarded. Pair codes solely rely on such complex components.

While Pair codes only need a linear number of Galois field operations for coding and decoding, they suffers from the coupon collector problem, such that the coding overhead is $\Theta(\log n)$. For small n it outperforms the other coding schemes, but for such cases systematic perfect erasure codes are still efficient enough. An example of such a systematic perfect efficient erasure resilient scheme with small number of XOR operations is the Circulant Cauchy code [23], where Boolean circulant matrices, also known as cyclotomic rings [25], have been used because of its efficiency. In such rings the word length is restricted to numbers w , where $w + 1$ is prime with 2 as a primitive root, i.e., all numbers $2^0, 2^1, \dots, 2^w$ modulo $w + 1$ are distinct. It is unknown how many such prime number exist, while they do appear to be quite often in prime numbers. This restrictions has been resolved in [23] by partitioning $w = \sum_i w_i$ such that each $w_i + 1$ is prime with 2 as a primitive root.

Here we drop the condition of $w + 1$ having 2 as a primitive root, since we observe that the unitary cyclotomic ring has some elements that can be inverted, even if it is not isomorphic to a finite field. So, we extend the set of word lengths being powers of two from $w \in \{2, 4\}$ to all numbers where $w + 1$ are Fermat primes, namely 2, 4, 16, 256, and 65 536. Still the partitioning technique works as well for all even w , such that Cyclone codes exist for all even w using $w_1 + w_2 = w$ if the Goldbach conjecture holds. Each read, write, addition, multiplication and division operation consist only of at most one cyclic shift operations and at most $w + 1$ bitwise XOR operations. It also allows the usage of the word

parallelism of processors, especially since simulations show that the coding overhead does not increase significantly for small w . Clearly, Cyclone codes can also be implemented using finite fields and general factors. However, we do not expect any significant benefit.

Outlook Raptor codes are based and on LT codes without any alternative. The situation has changed with the presentation of Cyclone codes. Since they outperform LT codes, it is straight-forward to look at a modified Raptor code based on Cyclone codes.

Another open area of research are special probability distributions optimized for Cyclone codes. At the moment we have only used the Ideal and Robust Soliton distribution, both of which are optimized for the single rule of Cyclone codes. However, the double rule harnesses the complex components of the random graph. Here lies the potential of even less coding overhead. Yet, the behavior of complex connected components in the 2-clause random graph, dynamically changed by the ripple effect of some Soliton distribution, is poorly understood. The investigation of such dynamic random graphs, where complex components are removed while edges are added to the residual graph, is crucial for improving Cyclone codes.

Acknowledgements We thank Christian Ortolfo for his valuable input and many fruitful discussions. Parts of this work have been supported by the *Sustainability Center Freiburg*.

References

- [1] Brittanney Amento, Martin Rötteler, and Rainer Steinwandt. Quantum binary field inversion: improved circuit depth via choice of basis representation. *arXiv preprint arXiv:1209.5491*, 2012.
- [2] Mario Blaum and Ron M. Roth. On lowest density MDS codes. *IEEE Transactions on Information Theory*, 45(1):46–59, 1999.
- [3] Johannes Blömer, Malik Kalfane, Marek Karpinski, Richard Karp, Michael Luby, and David Zuckerman. An XOR-based erasure-resilient coding scheme. ICSI Technical Report TR-95-048, ICSI, August 1995.
- [4] Béla Bollobás. The evolution of random graphs. *Transactions of the American Mathematical Society*, 286(1):257–274, 1984.
- [5] John W. Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. A digital fountain approach to reliable distribution of bulk data. In *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '98, pages 56–67, New York, NY, USA, 1998. ACM.
- [6] Ku-Young Chang, Dowon Hong, and Hyun-Sook Cho. Low complexity bit-parallel multiplier for $gf(2^m)$ defined by all-one polynomials using redundant representation. *Computers, IEEE Transactions on*, 54(12):1628–1630, 2005.
- [7] Jr. Costello, D.J. and Jr. Forney, G.D. Channel coding: The road to channel capacity. *Proceedings of the IEEE*, 95(6):1150–1177, June 2007.
- [8] Peter Elias. Coding for two noisy channels. In *Information Theory, Third London Symposium*, volume 67. London, England, 1955.
- [9] Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.
- [10] Willi Geiselmann, Jörn Müller-Quade, and Rainer Steinwandt. On “a new representation of elements of finite fields $GF(2^m)$ yielding small complexity arithmetic circuits”. *Computers, IEEE Transactions on*, 51(12):1460–1461, 2002.

- [11] David Hilbert. Mathematische probleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1900:253–297, 1900.
- [12] Esa Hyytiä, Tuomas Tirronen, and Jorma Virtamo. Optimizing the degree distribution of LT codes with an importance sampling approach. In *Proceedings of the 6th International Workshop on Rare Event Simulation*, pages 64–73, 2006.
- [13] Charanjit Jutla, Vijay Kumar, and Atri Rudra. On the circuit complexity of isomorphic galois field transformations. *Computer Science RC22652 (W0211-243), IBM Research Division, Thomas J. Watson Research Center, IBM Research Division Thomas J. Watson Research Center PO Box*, 218, 2002.
- [14] Jin Li. The efficient implementation of Reed-Solomon high rate erasure resilient codes. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 3, pages iii–1097. IEEE, 2005.
- [15] Michael Luby. LT codes. In *43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 271–280, 2002.
- [16] Michael Luby, Michael Mitzenmacher, Amin Shokrollahi, and Daniel Spielman. Efficient erasure correcting codes. *Information Theory, IEEE Transactions on*, 47(2):569–584, 2001.
- [17] Ghid Maatouk and Amin Shokrollahi. Analysis of the second moment of the lt decoder. *Information Theory, IEEE Transactions on*, 58(5):2558–2569, May 2012.
- [18] David J.C. MacKay. Fountain codes. *IEE Proceedings-Communications*, 152(6):1062–1068, 2005.
- [19] Ayan Mahalanobis. Are matrices useful in public-key cryptography? In *International Mathematical Forum*, [http://dx. doi. org/10.12988/imf](http://dx.doi.org/10.12988/imf), volume 8:39, pages 1939–1953, 2013.
- [20] Christian Ortolf, Christian Schindelhauer, and Arne Vater. Paircoding: Improving file sharing using sparse network codes. In *Internet and Web Applications and Services, 2009. ICIW'09. Fourth International Conference on*, pages 49–57. IEEE, 2009.
- [21] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.
- [22] Jörg Richstein. Verifying the goldbach conjecture up to $4 \cdot 10^{14}$. *Mathematics of computation*, 70(236):1745–1749, 2001.
- [23] Christian Schindelhauer and Christian Ortolf. Maximum distance separable codes based on circulant cauchy matrices. In Thomas Moscibroda and Adele A. Rescigno, editors, *Structural Information and Communication Complexity: 20th International Colloquium, SIROCCO 2013, Ischia, Italy, July 1-3, 2013, Revised Selected Papers*, pages 334–345, Cham, 2013. Springer International Publishing.
- [24] Amin Shokrollahi. Raptor codes. *Information Theory, IEEE Transactions on*, 52(6):2551–2567, 2006.
- [25] Joseph H Silverman. Fast multiplication in finite fields $GF(2^N)$. In *Cryptographic Hardware and Embedded Systems*, pages 122–134. Springer, 1999.
- [26] Huapeng Wu, M Anwar Hasan, Ian F Blake, and Shuhong Gao. Finite field multiplier using redundant representation. *Computers, IEEE Transactions on*, 51(11):1306–1316, 2002.

A Additional Simulation Results

Figures 4, 5, and 6 show the number of decoded data symbols (vertical axis), for a growing series of m code symbols for the above mentioned codes. The number of overall data symbols is $n = 10$ in Figure 4, $n = 1000$ in Fig. 5, and $n = 10\,000$ in Fig. 6. The word size is $w = 256$. For the Robust Soliton distribution we chose as parameters $\delta = 0.5$ and $c = 0.01$ [18]. The straight lines represent the average over 1000 tests. The dotted lines show the 90%-quantile of the set of decoded data symbols from 1000 runs with different random numbers.

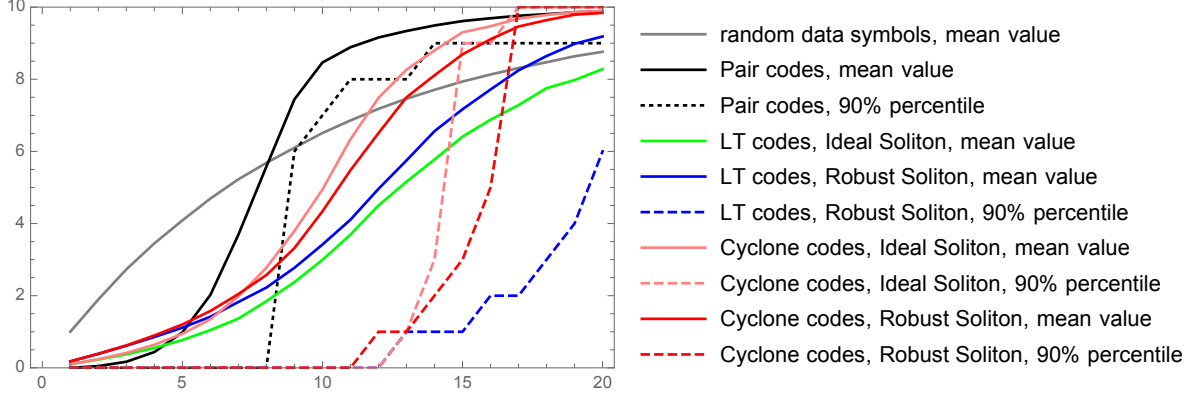


Figure 4: The average number and 90% quantile of the number of decoded symbols with respect to the number of available fountain codes for 10 data symbols and 1000 test samples.

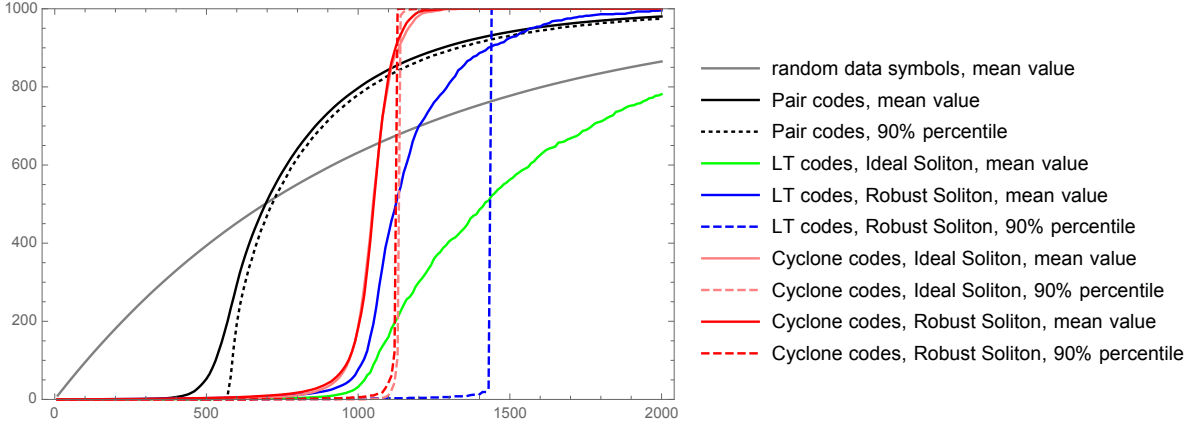


Figure 5: The average number and 90% quantile of the number of decoded symbols with respect to the number of available fountain codes for 1000 data symbols and 1000 test samples.

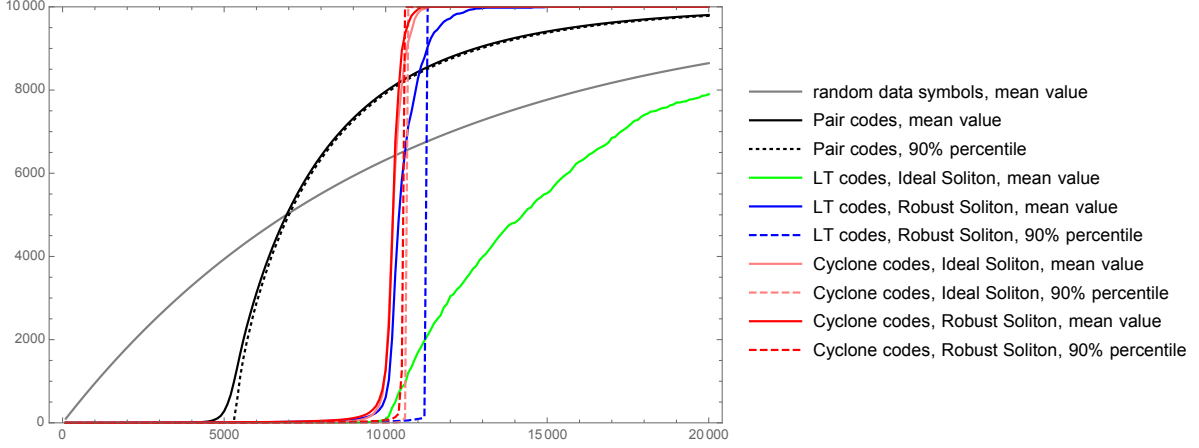


Figure 6: The average number and 90% quantile of the number of decoded symbols with respect to the number of available fountain codes for 10 000 data symbols and 1000 test samples.

Figure 7 compares different parameters for the Robust Soliton distribution, i.e., $\delta = 0.5$ and $c = 0.01$ and $c = 0.03$. The number of data symbols is $n = 2^i$ for $i = 1, \dots, 18$, i.e., $n = 2, 4, 8, \dots, 262\,144$, for tests. The test run increases the code size m until all n data symbols can be computed, then the overhead $m/n - 1$ is computed. The word size is $w = 256$. For random data symbols, Pair codes, LT codes with Robust Soliton distribution and Cyclone codes with Ideal Soliton and Robust Soliton distribution the average overhead is displayed in a log-log-plot.

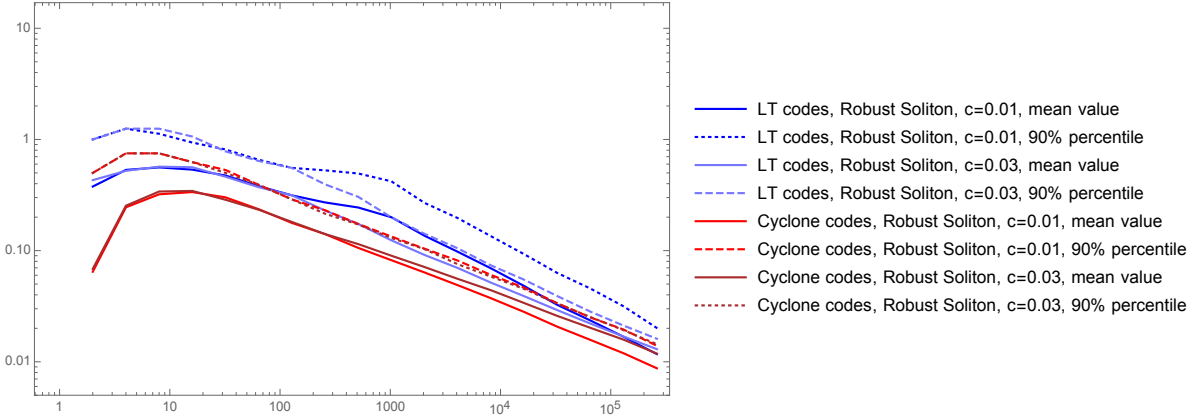


Figure 7: The average overhead of Cyclone and LT codes for increasing number of data symbols for $\delta = 0.5$ and $c \in \{0.01, 0.03\}$.

Figures 8, 9, 10, and 11 show histograms of the number of code symbols required to successfully decode $n = 10$, $n = 100$, $n = 1000$, and $n = 10\,000$ data symbols, respectively. Each histogram is the result of 1000 trials, where both LT codes and Cyclone codes use the Robust Soliton distribution with $\delta = 0.5$ and $c = 0.01$. The word size is $w = 256$. Given the same amount of code symbols, Cyclone codes are able to decode the complete set of data symbols significantly more often than LT codes.

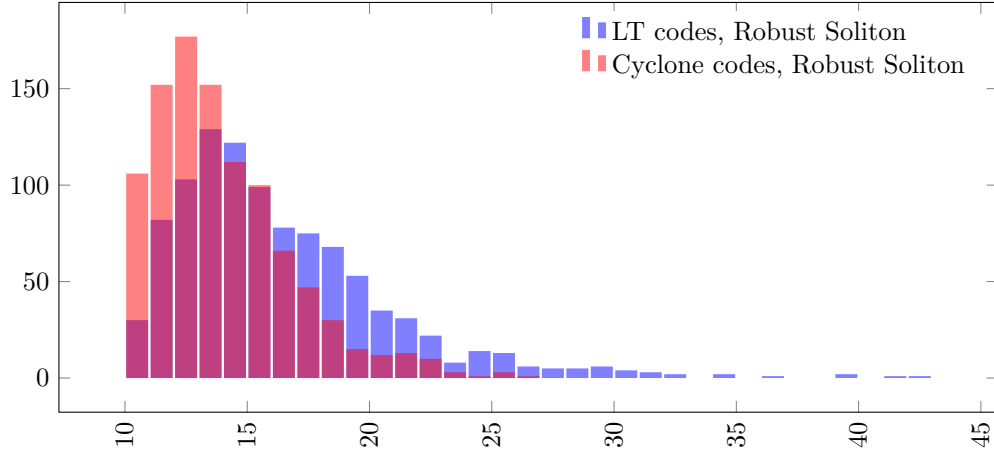


Figure 8: Histogram of the number of code symbols required to decode all $n = 10$ data symbols. The total number of trials is 1000 with parameters $\delta = 0.5$ and $c = 0.01$ for the Robust Soliton distribution.

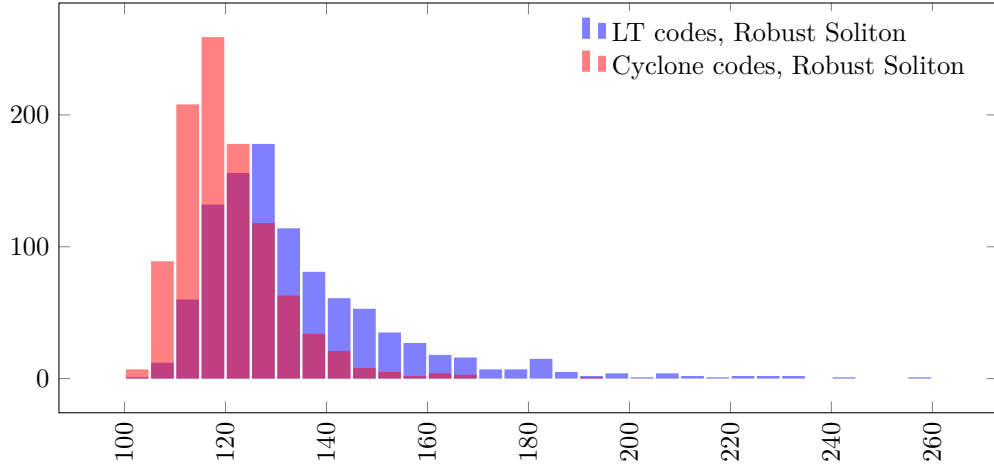


Figure 9: Histogram of the number of code symbols required to decode all $n = 100$ data symbols. The total number of trials is 1000 with parameters $\delta = 0.5$ and $c = 0.01$ for the Robust Soliton distribution.

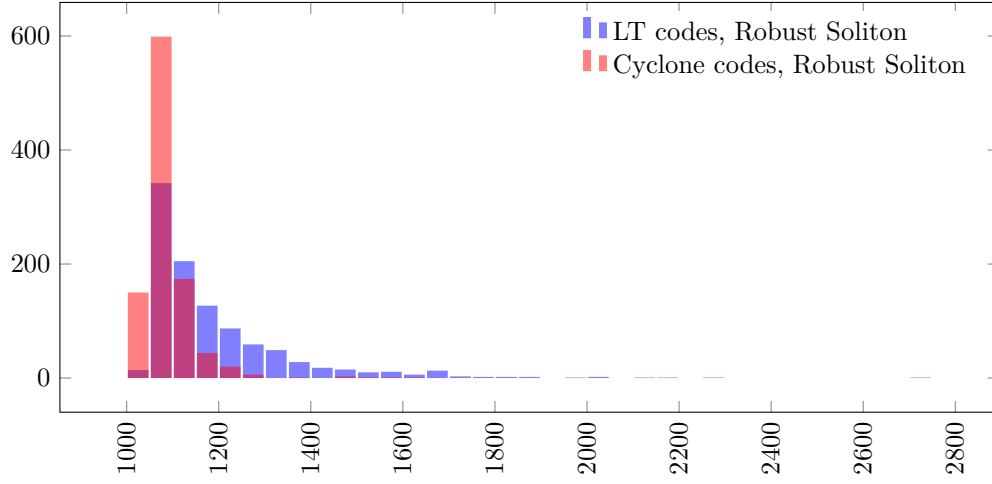


Figure 10: Histogram of the number of code symbols required to decode all $n = 1000$ data symbols. The total number of trials is 1000 with parameters $\delta = 0.5$ and $c = 0.01$ for the Robust Soliton distribution.

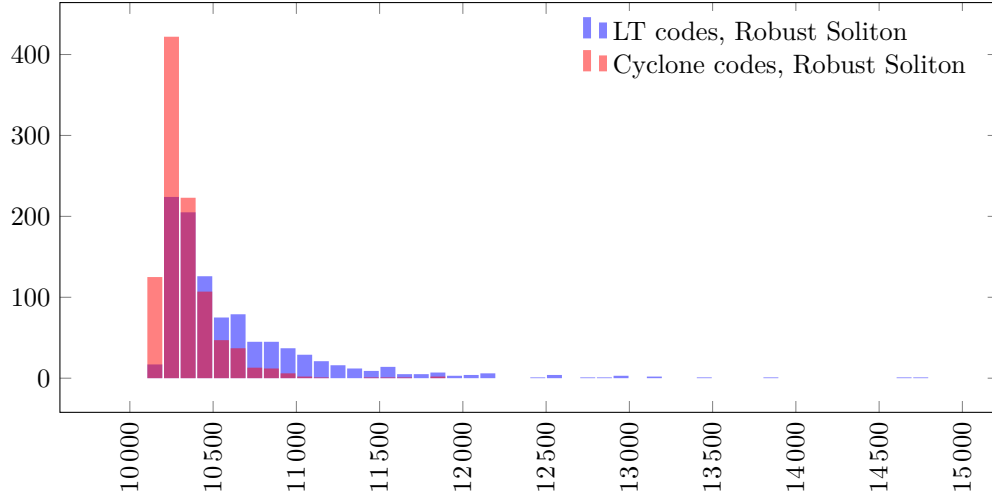


Figure 11: Histogram of the number of code symbols required to decode all $n = 10\,000$ data symbols. The total number of trials is 1000 with parameters $\delta = 0.5$ and $c = 0.01$ for the Robust Soliton distribution.